Program Listing for File gpu_resource_monitor.hpp

Return to documentation for file (include/holoscan/core/system/gpu_resource_monitor.hpp)

/* * SPDX-FileCopyrightText: Copyright (c) 2023 NVIDIA CORPORATION & AFFILIATES. All rights reserved. * SPDX-License-Identifier: Apache-2.0 * * Licensed under the Apache License, Version 2.0 (the "License"); * you may not use this file except in compliance with the License. * You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License. */ #ifndef HOLOSCAN_CORE_SYSTEM_GPU_RESOURCE_MONITOR_HPP #define HOLOSCAN_CORE_SYSTEM_GPU_RESOURCE_MONITOR_HPP #include <memory> #include <vector> #include "cuda_runtime_wrapper.h" #include "gpu_info.hpp" #include "nvml_wrapper.h" namespace holoscan { constexpr uint64_t kDefaultGpuMetrics = GPUMetricFlag::GPU_DEVICE_ID; class GPUResourceMonitor { public: explicit GPUResourceMonitor(uint64_t metric_flags = kDefaultGpuMetrics); virtual ~GPUResourceMonitor(); void init(); void close(); uint64_t metric_flags() const; void metric_flags(uint64_t metric_flags); GPUInfo update(uint32_t index, uint64_t metric_flags = GPUMetricFlag::DEFAULT); std::vector<GPUInfo> update(uint64_t metric_flags = GPUMetricFlag::DEFAULT); GPUInfo& update(uint32_t index, GPUInfo& gpu_info, uint64_t metric_flags = GPUMetricFlag::DEFAULT); GPUInfo gpu_info(uint32_t index, uint64_t metric_flags = GPUMetricFlag::DEFAULT); std::vector<GPUInfo> gpu_info(uint64_t metric_flags = GPUMetricFlag::DEFAULT); uint32_t num_gpus() const; bool is_integrated_gpu(uint32_t index); protected: bool bind_nvml_methods(); bool bind_cuda_runtime_methods(); bool init_nvml(); bool init_cuda_runtime(); void shutdown_nvml(); void shutdown_cuda_runtime(); void* handle_ = nullptr; void* cuda_handle_ = nullptr; *// NVML function pointers* nvml::nvmlErrorString_t nvmlErrorString = nullptr; nvml::nvmlInit_t nvmlInit = nullptr; nvml::nvmlDeviceGetCount_t nvmlDeviceGetCount = nullptr; nvml::nvmlDeviceGetHandleByIndex_t nvmlDeviceGetHandleByIndex = nullptr; nvml::nvmlDeviceGetHandleByPciBusId_t nvmlDeviceGetHandleByPciBusId = nullptr; nvml::nvmlDeviceGetHandleBySerial_t nvmlDeviceGetHandleBySerial = nullptr; nvml::nvmlDeviceGetHandleByUUID_t nvmlDeviceGetHandleByUUID =

```
nullptr; nvml::nvmlDeviceGetName_t nvmlDeviceGetName = nullptr;
nvml::nvmlDeviceGetIndex_t nvmlDeviceGetIndex = nullptr;
nvml::nvmlDeviceGetPciInfo_t nvmlDeviceGetPciInfo = nullptr;
nvml::nvmlDeviceGetSerial_t nvmlDeviceGetSerial = nullptr;
nvml::nvmlDeviceGetUUID_t nvmlDeviceGetUUID = nullptr;
nvml::nvmlDeviceGetMemoryInfo_t nvmlDeviceGetMemoryInfo = nullptr;
nvml::nvmlDeviceGetUtilizationRates_t nvmlDeviceGetUtilizationRates = nullptr;
nvml::nvmlDeviceGetPowerManagementLimit_t
nvmlDeviceGetPowerManagementLimit = nullptr;
nvml::nvmlDeviceGetPowerUsage_t nvmlDeviceGetPowerUsage = nullptr;
nvml::nvmlDeviceGetTemperature_t nvmlDeviceGetTemperature = nullptr;
nvml::nvmlShutdown_t nvmlShutdown = nullptr; // CUDA Runtime function pointers
cuda::cudaGetErrorString_t cudaGetErrorString = nullptr;
cuda::cudaGetDeviceCount_t cudaGetDeviceCount = nullptr;
cuda::cudaGetDeviceProperties_t cudaGetDeviceProperties = nullptr;
cuda::cudaDeviceGetPCIBusId_t cudaDeviceGetPCIBusId = nullptr;
cuda::cudaMemGetInfo_t cudaMemGetInfo = nullptr; uint64_t metric_flags_ =
kDefaultGpuMetrics; bool is_cached_ = false; uint32_t gpu_count_ = 0;
std::vector<GPUInfo> gpu_info_; std::vector<nvml::nvmlDevice_t> nvml_devices_; }; }
// namespace holoscan #endif/*
HOLOSCAN_CORE_SYSTEM_GPU_RESOURCE_MONITOR_HPP */
```